



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

Ogasawara et al.

Serial No.: 10/787,005

Filed: February 25, 2004

For: COMPILER DEVICE, PROGRAM, AND RECORDING MEDIUM

Date: December 7, 2004

Group Art Unit: 2655

Examiner: Not yet assigned

Docket No.: JP920030021US1

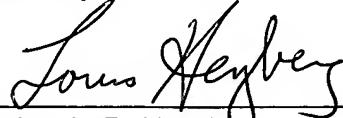
Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMISSION OF PRIORITY DOCUMENT

Sir:

Enclosed herewith is a certified copy of Japanese Application No. 2003-49414
filed February 26, 2003, in support of applicant's claim to priority under 35 U.S.C. 119.

Respectfully submitted,

By 

Louis P. Herzberg
Reg. No. 41,500
Phone No. (914) 945-2885

IBM Corporation
Intellectual Property Law Dept.
P.O. Box 218
Yorktown Heights, NY 10598

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2003年 2月26日

出 願 番 号

Application Number:

特願2003-049414

ST.10/C]:

[JP2003-049414]

出 願 人

Applicant(s):

インターナショナル・ビジネス・マシーンス・コーポレーション

2003年 5月20日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田 信一郎

出証番号 出証特2003-3037028

【書類名】 特許願

【整理番号】 JP9030021

【提出日】 平成15年 2月26日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/45

【発明者】

 【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

 【氏名】 竹内 幹雄

【発明者】

 【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

 【氏名】 小野寺 民也

【発明者】

 【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

 【氏名】 小笠原 武史

【特許出願人】

 【識別番号】 390009531

 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

 【識別番号】 100086243

 【弁理士】

 【氏名又は名称】 坂口 博

【代理人】

 【識別番号】 100091568

 【弁理士】

 【氏名又は名称】 市位 嘉宏

【代理人】

【識別番号】 100108501

【弁理士】

【氏名又は名称】 上野 剛史

【復代理人】

【識別番号】 100104156

【弁理士】

【氏名又は名称】 龍華 明裕

【手数料の表示】

【予納台帳番号】 053394

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9706050

【包括委任状番号】 9704733

【包括委任状番号】 0207860

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンパイラ装置、コンパイラプログラム、及び記録媒体

【特許請求の範囲】

【請求項 1】 文字列を操作するプログラムを最適化するコンパイラ装置であって、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

【請求項 2】 前記複数の追加命令によって文字列を追加した後に初めて前記文字列変数を参照する参照命令を検出する参照命令検出部を更に備え、

前記追加コード生成部は、前記追加コードを、全ての前記格納コードより後、かつ前記参照命令より先に実行される個所に生成する請求項 1 記載のコンパイラ装置。

【請求項 3】 前記追加命令検出部は、前記追加命令として、文字列を追加する処理が許可されていない不変文字列変数を、文字列を追加する処理が許可された可変文字列変数に変換する命令と、前記可変文字列変数に前記追加文字列を追加する命令と、前記可変文字列変数を不変文字列変数に変換する命令との組合せを検出する請求項 1 記載のコンパイラ装置。

【請求項 4】 文字列を操作するプログラムを最適化するコンパイラ装置であって、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する

複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

【請求項 5】 文字列を操作するプログラムを最適化するコンパイラ装置であって、

文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変命令を検出する不変命令検出部と、

前記不変文字列変数を可変文字列変数に変換する可変命令を検出する可変命令検出部と、

前記不変命令から前記可変命令までの間に実行される命令が、前記不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変命令から前記可変命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変命令の変換元の可変文字列変数と、前記可変命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変命令を除去し、前記可変命令より後で、前記不変命令の変換元となる可変文字列変数を、前記可変命令により変換された可変文字列変数として使用させる命令除去部とを備えるコンパイラ装置。

【請求項 6】 前記不変文字列変数に格納された文字列が参照されない場合に、前記命令除去部は、前記不変命令を更に除去する請求項 5 記載のコンパイラ装置。

【請求項 7】 前記命令除去部は、前記不変命令を、当該不変命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変命令の変換先である不変文字列変数に格納されている

文字列が参照されない場合に、当該不変命令を除去する部分不用代入文除去を行う請求項 6 記載のコンパイラ装置。

【請求項 8】 前記可変命令検出部は、可変文字列変数として用いられる記憶領域を確保する命令と、当該可変文字列変数に、前記不変文字列変数に格納された文字列を追加する命令との組合せを、前記可変命令として検出する請求項 5 記載のコンパイラ装置。

【請求項 9】 前記可変命令検出部により検出された前記可変命令を、当該可変命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う部分冗長除去部を更に備え、

前記部分冗長性の前記除去処理が実行されたプログラムにおいて、前記不変命令及び前記可変命令の間に実行される命令が、前記不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変命令から前記可変命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変命令の変換元の可変文字列変数と、前記可変命令により変換された可変文字列変数の何れも書き換えない場合に、前記命令除去部は、前記可変命令を除去する請求項 5 記載のコンパイラ装置。

【請求項 10】 前記命令除去部は、前記不変命令を、当該不変命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変命令を除去する部分不用代入文除去を行う請求項 9 記載のコンパイラ装置。

【請求項 11】 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と

して機能させるコンパイラプログラム。

【請求項 1 2】 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と

して機能させるコンパイラプログラム。

【請求項 1 3】 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変命令を検出する不変命令検出部と、

前記不変文字列変数を可変文字列変数に変換する可変命令を検出する可変命令検出部と、

前記不変命令から前記可変命令までの間に実行される命令が、前記不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変命令から前記可変命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変命令の変換元の可変文字列変数と、前記可変命令によ

り変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部として機能させるコンパイラプログラム。

【請求項 1 4】 請求項 1 1 から 1 3 の何れかに記載のコンパイラプログラムを記録した記録媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、コンパイラ装置、コンパイラプログラム、及び記録媒体に関する。特に本発明は、文字列を操作するプログラムを最適化するコンパイラ装置、コンパイラプログラム、及び記録媒体に関する。

【0 0 0 2】

【従来の技術】

近年、プログラムの保守性や堅牢性を高めるために、プログラマにとって直感的な演算子を記載することのできるプログラム言語が普及している。例えば、J a v a（登録商標）言語において、プログラマは、文字列変数に文字列を追加する処理を、「+」演算子により、簡易に記載することができる。

また、従来、コンパイラにおいて、部分冗長性を除去する技術（非特許文献 1 参照。）及び部分不用代入文を除去する技術（非特許文献 2 参照。）が用いられている。

【0 0 0 3】

【非特許文献 1】

J. K n o o p、O. R u t h i n g a n d B. S t e f f e n、L a z y c o d e m o t i o n、I n P L D I '92、p. 2 2 4 - 2 3 4、1 9 9 2. 邦題「L a z yコード移動」

【0 0 0 4】

【非特許文献 2】

J. Knoop, O. Ruthing and B. Steffen
、Partial dead code elimination、In PLDI '94、p. 147-158、1994. 邦題「部分不用代入文除去」

【0005】

【発明が解決しようとする課題】

文字列の追加処理は、実際には、メモリ領域の確保及びデータの複写等の、多数の処理の集まりにコンパイルされる。しかしながら、同一の文字列変数に、複数の文字列が順次追加される場合には、メモリ領域の確保及びデータの複写等が冗長となる場合がある。更に、これらの冗長な命令は、部分冗長性の除去及び部分不用代入文の除去の技術をそのまま適用しても、適切に除去されない場合がある。例えば、サーバ用プログラムにおいて、文字列の追加処理が頻繁に用いられる場合には、サーバの動作効率を低めてしまっていた。

そこで本発明は、上記の課題を解決することのできるコンパイラ装置、コンパイラプログラム、及び記録媒体を提供することを目的とする。この目的は特許請求の範囲における独立項に記載の特徴の組み合わせにより達成される。また従属項は本発明の更なる有利な具体例を規定する。

【0006】

【課題を解決するための手段】

即ち、本発明の第1の形態によると、文字列を操作するプログラムを最適化するコンパイラ装置であって、プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、プログラムにおける、文字列変数を参照する命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置、コンパイラプログラム、及びコンパイラプログラムを記録した記録媒体を提供する。

なお上記の発明の概要は、本発明の必要な特徴の全てを列挙したものではなく

、これらの特徴群のサブコンビネーションも又発明となりうる。

【0007】

【発明の実施の形態】

以下、発明の実施の形態を通じて本発明を説明するが、以下の実施形態は特許請求の範囲にかかる発明を限定するものではなく、又実施形態の中で説明されている特徴の組み合わせの全てが発明の解決手段に必須であるとは限らない。

【0008】

(第1実施形態)

図1は、コンパイラ装置10のブロック図を示す。コンパイラ装置10は、例えば、Java（登録商標）のJust-In-Timeコンパイラであり、文字列を格納する文字列変数に文字列を追加する追加命令を最適化することを目的とする。コンパイラ装置10は、追加命令を検出する追加命令検出部100と、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部110と、文字列変数を参照する参照命令を検出する参照命令検出部120と、バッファに格納された追加文字列を文字列変数に追加する追加コード生成部130とを備える。

【0009】

追加命令検出部100は、コンパイル対象のプログラムを取得すると、当該プログラムにおいて、同一の文字列変数に文字列を追加する複数の追加命令を検出し、検出結果を格納コード生成部110及び参照命令検出部120に送る。格納コード生成部110は、追加命令の検出結果に基づき、コンパイル対象のプログラムにおける、追加命令検出部100によって検出された複数の追加命令のそれぞれに代えて、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する。そして、格納コード生成部110は、格納コードが生成された当該プログラムを追加コード生成部130に送る。

【0010】

参照命令検出部120は、追加命令の検出結果に基づき、コンパイル対象のプログラムにおいて、複数の追加命令によって文字列を追加した後初めて文字列変数を参照する参照命令を検出し、検出結果を追加コード生成部130に送る。追

加コード生成部 1 3 0 は、参照命令の検出結果に基づき、格納コード生成部 1 1 0 から受け取ったプログラムにおける、全ての格納コードより後、かつ参照命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。そして、追加コード生成部 1 3 0 は、追加コードが生成された当該プログラムを、コンパイル結果のプログラムとして出力する。なお、コンパイラ装置 1 0 は、本図の各部材が行う処理より先又は後に、他の最適化を行ってもよい。

【0 0 1 1】

図 2 は、コンパイラ装置 1 0 の動作フローを示す。コンパイラ装置 1 0 は、例えば、J a v a（登録商標）における、コンパイル対象のプログラムを含むクラスファイルを取得する（S 2 0 0）。そして、追加命令検出部 1 0 0 は、同一の文字列変数に文字列を追加する複数の追加命令を検出する（S 2 1 0）。格納コード生成部 1 1 0 は、コンパイル対象のプログラムにおいて、追加命令検出部 1 0 0 において検出された複数の追加命令のそれぞれに代えて、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する（S 2 2 0）。参照命令検出部 1 2 0 は、複数の追加命令によって文字列を追加した後に初めて文字列変数を参照する参照命令を検出する（S 2 3 0）。そして、追加コード生成部 1 3 0 は、全ての格納コードより後、かつ参照命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する（S 2 4 0）。

【0 0 1 2】

図 3（a）は、最適化対象のプログラムのソースコードの一例を示す。1 行目の命令は、文字列変数の一例である s に、空の文字列 “ ” を代入する命令である。2 から 4 行目の命令のそれぞれは、変数 c に格納された追加文字列を s と連結（concatenate）し、その結果を s に代入する命令、即ち、変数 c に格納された追加文字列を s に追加する追加命令である。なお、2 から 4 行目の命令のそれぞれは、ループにより繰り返し実行される、ソースコード上で同一の追加命令であってもよい。5 行目の命令は、2 から 4 行目のそれぞれの追加命令によって文字列を追加した後に初めて s を参照する参照命令である。

【 0 0 1 3 】

図 3 (b) は、プログラムがコンパイルされた結果を示す。より詳細には、例えば、本図は、J a v a (登録商標)におけるバイトコードコンパイラが、図 3 (a) に示したソースコードを、バイトコードにコンパイルした結果の一例を示す。なお、実際には、バイトコードコンパイラは、各命令を数値データで示したバイトコードを出力するが、本図は、説明の都合上バイトコードの意味を示す擬似コードを示す。なお、以降に説明する図 5 及び図 8 から 1 2 においても、説明の都合上、命令の意味を示す擬似コードを用いる。

【 0 0 1 4 】

1 行目の命令は、文字列の追加が許可されていない不変文字列変数 *s* を、文字列の追加が許可された可変文字列変数 *s b* に変換する命令であり、例えば、文字列の追加が許可された可変文字列変数 *s b* を格納する領域をメモリに確保し、文字列の追加が許可されていない不変文字列変数 *s* に格納される文字列を、*s b* に複写する。2 行目の命令は、*s b* に、変数 *c* に格納された追加文字列を追加する命令である。なお、変数 *c* がキャラクタ変数でない場合には、2 行目の命令に加え、更に、*valueOf* メソッドが実行され、その過程で不変文字列変数が更に確保される場合がある。3 行目の命令は、*s b* を、不変文字列 *s* に変換する命令である。以降同様に、4 行目及び 7 行目の命令のそれぞれは、1 行目の命令と略同一であり、5 行目及び 8 行目の命令のそれぞれは、2 行目の命令と略同一であり、6 行目及び 9 行目の命令のそれぞれは、3 行目の命令と略同一である。また、1 0 行目の命令は、不変文字列変数 *s* を参照する命令である。

【 0 0 1 5 】

このように、不変文字列変数 *s* に、新たに文字列 *c* を追加する命令は、メモリ上の領域確保及びデータの複写を含む複数の命令にコンパイルされる。本図のプログラムによると、最終的に用いられない変数の領域確保が、ループで繰り返されてしまう。例えば、1 行目、4 行目、及び 7 行目において、最終的に不要な変数 *s b* の領域確保が繰り返される。更に、新たに確保されたメモリ領域を初期化する処理が必要となり、効率が悪い。また、コンパイラ装置 1 0 が動的メモリ管理を行う場合には、不要なメモリ領域は、ガーベージコレクション (G a r b a

ge Collection) により頻繁に開放され、効率が悪い。

【0016】

ここで、文字列を追加する処理が許可された可変文字列変数 (mutable string variable) とは、プログラム言語の仕様により、文字列の追加操作が許可されている変数であり、一例としては、Java (登録商標) 言語における StringBuffer オブジェクトのインスタンスである。そして、プログラム言語がオブジェクト指向言語の場合、可変文字列変数を含む可変文字列オブジェクトは、可変文字列変数に格納される文字列に新たに文字列を追加する追加メソッド、例えば、append メソッドを有していてもよい。

【0017】

一方、文字列を追加する処理が許可されていない不変文字列変数 (immutable string variable) とは、プログラム言語の仕様により、文字列の追加操作が許可されていない変数であり、一例としては、Java (登録商標) 言語における String オブジェクトのインスタンスである。そして、コンパイラ装置 10 は、不変文字列変数に格納された文字列が変化しないと仮定してコンパイル処理を行えるので、効率的となる。例えば、コンパイラ装置 10 は、不変文字列変数へのアクセスに、排他制御を必要としないので、高速に実行させることができる。

【0018】

図 3 (c) は、コンパイラ装置 10 による最適化を終えたプログラムの一例を示す。追加命令検出部 100 は、追加命令として、不変文字列変数 s を可変文字列変数 s b に変換する図 3 (b) の 1 行目の命令と、可変文字列変数 s b に追加文字列 c を追加する図 3 (b) の 2 行目の命令と、可変文字列変数 s b を不変文字列変数 s に変換する図 3 (b) の 3 行目の命令との組合せを検出する。そして、格納コード生成部 110 は、追加命令に代えて、追加文字列 c のデータをバッファに格納する格納コード、例えば、図 3 (c) の 2 行目の命令を生成する。

【0019】

ここで、バッファとは、プログラムを実行するスレッド毎に、予め確保されたメモリ上の領域である。バッファは、本図のプログラムの実行に加え、同一のス

レッドによる他の処理に用いられるので、文字のデータサイズより充分に大きいことが望ましい。一例としては、バッファのサイズは、1 M b y t e であってもよいし、1 0 M b y t e であってもよい。また、本図で示した変数 b u f は、バッファにおける空き領域の先頭のアドレスを格納する変数である。

【 0 0 2 0 】

追加コード生成部 1 3 0 は、不変文字列変数 s を参照する 7 行目の命令より先に実行される箇所、例えば、5 行目に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。例えば、追加コード生成部 1 3 0 は、追加コードとして、追加文字列を格納しているバッファのアドレスを、不変文字列変数を含むオブジェクトのインスタンス変数 (s . v a l u e) に代入する命令を生成する。この結果、7 行目の命令は、追加処理を終えた後の文字列変数を適切に参照できる。更に、本図に示したプログラムを実行するスレッドが、本図に示したプログラム以降に更に他のプログラムを実行する場合に備え、追加コード生成部 1 3 0 は、バッファの空き領域を示す情報を更新する処理を行うコードを生成することが望ましい。例えば、追加コード生成部 1 3 0 は、変数 b u f の値を更新する命令を 6 行目に生成することにより、後に実行されるコードに、バッファの空き領域を適切に知らせることができる。

【 0 0 2 1 】

このように、コンパイラ装置 1 0 は、可変文字列変数の領域をメモリに確保する命令及び文字列のデータを複写する命令を除去することにより、文字列を追加する追加命令を最適化することができる。

なお、本実施形態において、コンパイラ装置 1 0 は、例えば、バイトコードを実行コードにコンパイルする J u s t - I n - T i m e コンパイラである。これに代えて、コンパイラ装置 1 0 は、ソースプログラムをバイトコードにコンパイルするバイトコードコンパイラであってもよい。この場合、追加命令検出部 1 0 0 は、ソースプログラムにおいて、文字列変数に文字列を追加する「+」演算子を、追加命令として検出する。そして、格納コード生成部 1 1 0 は、「+」演算子に代えて、格納コードを生成する。即ち、コンパイラ装置 1 0 は、図 3 (a) に示したソースプログラムを入力とし、図 3 (b) のプログラムを介さずに、図

3 (c) と略同一の処理を行うバイトコードを出力してもよい。

【 0 0 2 2 】

図 4 は、変形例において、コンパイラ装置 1 0 がプログラムを最適化する一例を示す。本例におけるコンパイラ装置 1 0 は、図 3 (c) に示したプログラムに代えて、本図に示したプログラムを生成してもよい。

【 0 0 2 3 】

コンパイラ装置 1 0 は、バッファを初期化する 1 行目の命令を生成する。バッファ `b u f` は、連結リスト構造であり、連結リストの各要素は、追加文字列を 1 ずつ格納することができる。格納コード生成部 1 1 0 は、図 3 (a) における追加命令に代えて、追加文字列 `c` のデータをバッファに格納する図 4 の 2 行目の命令と、連結リストにおける次の要素を示すアドレスを取得する 3 行目の命令とを、格納コードとして生成する。以降同様に、格納コード生成部 1 1 0 は、複数の追加命令のそれぞれに代えて、4 行目及び 6 行目において、格納コードを生成する。そして、追加コード生成部 1 3 0 は、不変文字列変数 `s` を参照する 9 行目の命令より先に実行される個所、例えば、8 行目に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。

【 0 0 2 4 】

このように、格納コード生成部 1 1 0 は、複数の追加命令のそれぞれに代えて、追加文字列及び追加文字列が格納されているアドレスとの組合せを、連結リスト状に、バッファに格納してもよい。この場合、連結リストをヌルポインタで予め終端しておくことで、コンパイラ装置 1 0 は、メモリアクセス違反を検出することにより、連結リストの終端を検出できる。これにより、各格納コードにおける、連結リストの終端チェック命令が省略できるので、処理を高速化することができる。

【 0 0 2 5 】

図 5 (a) は、第 1 の他の方法においてプログラムが最適化された結果を示す。より詳細には、本例は、熟練した `J a v a` (登録商標) プログラマが、文字列操作の実行効率を高めるべく、「+」演算子を用いることなく、文字列の追加処理を記述した例である。1 行目の命令は、可変文字列変数 `s b` を格納する領域を

メモリに確保し、不変文字列変数 `s` に格納される文字列を、`s b` に複写する命令である。2 行目の命令は、`s b` に、変数 `c` に格納された追加文字列を追加する命令である。以降同様に、3 行目及び 4 行目の命令のそれぞれは、2 行目の命令と略同一である。そして、5 行目の命令は、`s b` を、不変文字列 `s` に変換する命令である。また、6 行目の命令は、不変文字列変数 `s` を参照する命令である。

【0026】

このように、本例によると、追加命令に代えて、可変文字列変数に文字列を追加するコードを用いるので、図 3（b）に示したプログラムより性能が高い。

【0027】

図 5（b）は、第 2 の他の方法においてプログラムが最適化された結果を示す。本例において、不変文字列変数を含む `String` オブジェクトは、`Java`（登録商標）の言語仕様に規定されていない、非公開文字列構築子を有している。非公開文字列構築子によると、新たな `String` オブジェクトを生成すると共に、新たな当該 `String` オブジェクトに、複数の `String` オブジェクトを連結した文字列を複写することができる。例えば、2 から 4 行目の命令のそれぞれは、新たに不変文字列変数を生成し、当該不変文字列変数に、2 つの不変文字列変数の内容を複写する命令である。

【0028】

これにより、可変文字列変数への変換処理を省くことができるので、本例に示したプログラムは、図 3（b）に示したプログラムより性能が高い。しかしながら、2 から 4 行目のそれぞれにおいて、不変文字列変数を格納する領域が確保されるにも関わらず、5 行目の命令においては、4 行目の命令により生成された不変文字列変数のみを参照するので、依然として冗長である。

【0029】

これに対し、第 1 実施形態におけるコンパイラ装置 10 は、冗長なメモリ確保命令や変換処理を除去することにより、文字列を操作する命令を適切に最適化することができる。

【0030】

（第 2 実施形態）

図 6 は、コンパイラ装置 6 0 のブロック図を示す。コンパイラ装置 6 0 は、例えば、J a v a（登録商標）の J u s t - I n - T i m e コンパイラであり、文字列を格納する文字列変数に文字列を追加する追加命令を最適化することを目的とする。コンパイラ装置 6 0 は、不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部の一例である意味復元部 6 0 0 と、部分冗長性の除去処理を実行する部分冗長除去部 6 1 0 と、可変文字列変数を不変文字列変数に変換する不変化命令を検出する不変化命令検出部 6 2 0 と、可変化命令を除去する命令除去部 6 3 0 と、検出した可変化命令を実行可能な状態に変換する逆意味復元部 6 4 0 と、コンパイル対象のプログラムを実行することによりプロファイルデータを取得する計測コード付きプログラム実行部 6 4 2 と、プロファイルデータを格納する不変文字列変数最終長データベース 6 4 5 と、可変文字列変数の初期長を指定する可変文字列変数初期長指定部 6 5 0 とを備える。

【 0 0 3 1 】

意味復元部 6 0 0 は、コンパイル対象のプログラムを取得すると、不変文字列変数を可変文字列変数に変換する可変化命令を検出する。より詳細には、例えば、意味復元部 6 0 0 は、可変化命令の機能を実現する複数の命令の組合せを検出し、当該命令の組合せを、可変化命令を示す擬似命令に変換する。そして、意味復元部 6 0 0 は、変換した擬似命令を含むプログラムを、部分冗長除去部 6 1 0 に送る。

【 0 0 3 2 】

部分冗長除去部 6 1 0 は、擬似命令を含むプログラムを意味復元部 6 0 0 から受け取ると、意味復元部 6 0 0 により検出された可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う。なお、移動とは、受け取ったプログラムにおける可変化命令を削除し、当該可変化命令を制御フローの合流元に挿入することにより、命令の実行順序を変更することである。そして、部分冗長除去部 6 1 0 は、部分冗長性の除去処理を行ったプログラムを不変化命令検出部 6 2 0 に送る。

【 0 0 3 3 】

不変化命令検出部 6 2 0 は、部分冗長除去部 6 1 0 から受け取ったプログラム

において、可変文字列変数を不変文字列変数に変換する不変命令を検出する。そして、不変命令検出部 6 2 0 は、部分冗長除去部 6 1 0 から受け取ったプログラムを、検出結果と共に、命令除去部 6 3 0 に送る。

【 0 0 3 4 】

命令除去部 6 3 0 は、代数簡約部 6 3 4 と、部分不用代入文除去部 6 3 8 を有する。代数簡約部 6 3 4 は、不変命令検出部 6 2 0 からプログラムを受け取ると、代数簡約的手法により、可変命令を除去する。例えば、代数簡約部 6 3 4 は、「可変文字列変数 t を不変文字列変数に変換した結果が s であるならば、不変文字列変数 s を変換した結果は t になる」という代数的性質を用いて、可変命令を除去する。具体的には、代数簡約部 6 3 4 は、不変命令から可変命令までの間に実行される命令が、不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ可変命令から可変命令により変換された可変文字列変数の使用の間に実行される命令が、不変命令の変換元の可変文字列変数と、可変命令により変換された可変文字列変数の何れも書き換えない場合に、可変命令を除去する。そして、代数簡約部 6 3 4 は、可変命令より後で、不変命令の変換元となる可変文字列変数を、可変命令により変換された可変文字列変数として使用させる。その後、代数簡約部 6 3 4 は、可変命令を除去したプログラムを部分不用代入文除去部 6 3 8 に送る。

【 0 0 3 5 】

部分不用代入文除去部 6 3 8 は、部分不用代入文除去の処理として、以下の処理を行う。部分不用代入文除去部 6 3 8 は、不変命令を、当該不変命令より後に実行される分岐命令の分岐先のそれぞれに移動する。そして、部分不用代入文除去部 6 3 8 は、当該分岐命令の分岐先のそれぞれにおいて、不変命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変命令を除去する。そして、部分不用代入文除去部 6 3 8 は、部分不用代入文除去の処理を行ったプログラムを、逆意味復元部 6 4 0 に送る。

【 0 0 3 6 】

逆意味復元部 6 4 0 は、不変命令検出部 6 2 0 により変換された不変命令を、不変命令の機能を実現する複数の命令の組合せに戻す変換処理を行う。そ

して、逆意味復元部 6 4 0 は、変換結果のプログラムを可変文字列変数初期長指定部 6 5 0 に送る。

【 0 0 3 7 】

計測コード付きプログラム実行部 6 4 2 は、コンパイラ装置 1 0 が行うコンパイルの効率を高めるべく、コンパイル対象のプログラムを予め実行し、実行の状態を計測する。例えば、計測コード付きプログラム実行部 6 4 2 は、文字列変数に実際に格納された文字列の長さを計測し、計測結果を不変文字列変数最終長データベース 6 4 5 に格納する。

【 0 0 3 8 】

可変文字列変数初期長指定部 6 5 0 は、コンパイル対象のプログラムを逆意味復元部 6 4 0 から受け取り、可変文字列変数に実際に格納された文字列の長さを示すサイズ情報を不変文字列変数最終長データベース 6 4 5 から受け取る。そして、可変文字列変数初期長指定部 6 5 0 は、可変文字列変数が新たに生成される場合には、サイズ情報が示す長さの領域を予め確保するように、可変文字列変数の初期長を設定する。これにより、コンパイラ装置 1 0 は、プログラムの実行時に、可変文字列変数のメモリ領域を伸張させることを防ぐことができる。そして、可変文字列変数初期長指定部 6 5 0 は、可変文字列変数の初期長を設定したプログラムを、コンパイル結果のプログラムとして出力する。

なお、コンパイラ装置 6 0 は、計測コード付きプログラム実行部 6 4 2 と、不変文字列変数最終長データベース 6 4 5 と、可変文字列変数初期長指定部 6 5 0 とを備えなくともよい。この場合、コンパイラ装置 6 0 は、コンパイラ装置 6 0 の設計者により予め設定された値を、可変文字列変数の初期長としてもよい。

【 0 0 3 9 】

図 7 は、コンパイラ装置 6 0 の動作フローを示す。意味復元部 6 0 0 は、可変化命令の機能を実現する複数の命令の組合せを検出し、当該命令の組合せを、可変化命令を示す擬似命令に変換する意味復元を行うことにより、不変文字列変数を可変文字列変数に変換する可変化命令を検出する（S 7 0 0）。部分冗長除去部 6 1 0 は、擬似命令を含むプログラムを意味復元部 6 0 0 から受け取ると、意味復元部 6 0 0 により検出された可変化命令を、当該可変化命令より前において

合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う（S 7 1 0）。不変命令検出部 6 2 0 は、部分冗長除去部 6 1 0 から受け取ったプログラムにおいて、可変文字列変数を不変文字列変数に変換する不変命令を検出する（S 7 2 0）。

【 0 0 4 0 】

代数簡約部 6 3 4 は、代数簡約的手法により、可変命令を除去する。例えば、代数簡約部 6 3 4 は、不変命令から可変命令までの間に実行される命令が、不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ可変命令から可変命令により変換された可変文字列変数の使用の間に実行される命令が、不変命令の変換元の可変文字列変数と、可変命令により変換された可変文字列変数の何れも書き換えない場合に、可変命令を除去する。そして、代数簡約部 6 3 4 は、不変命令の変換元となる可変文字列変数を、除去した可変命令が存在していた個所より後の部分で、可変命令により変換された可変文字列変数として使用させる（S 7 3 0）。

【 0 0 4 1 】

部分不用代入文除去部 6 3 8 は、不変命令を、当該不変命令より後に実行される分岐命令の分岐先のそれぞれにおいて実行させるように、命令の実行順序を変更する。そして、部分不用代入文除去部 6 3 8 は、当該分岐命令の分岐先のそれぞれにおいて、不変命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変命令を除去する部分不用代入文除去の処理を行う（S 7 4 0）。

【 0 0 4 2 】

逆意味復元部 6 4 0 は、意味復元部 6 0 0 により変換された可変命令を、可変命令の機能を実現する複数の命令の組合せに戻す変換処理を行う（S 7 5 0）。可変文字列変数初期長指定部 6 5 0 は、計測コード付プログラム実行部 6 4 2 により予め計測され、可変文字列変数に実際に格納された文字列の長さを示すサイズ情報に基づき、可変文字列変数の初期長を設定する（S 7 6 0）。

【 0 0 4 3 】

図 8 は、コンパイラ装置 6 0 が最適化する対象のプログラムの一例を示す。本

図のプログラムは、1 から 3 行目の命令を含む基本ブロック 8 0 0 と、4 から 9 行目の命令を含む基本ブロック 8 1 0 と、1 0 行目の命令を含む基本ブロック 8 2 0 とを有する。

【0 0 4 4】

1 行目の命令は、不変文字列変数の一例である `s` に、空の文字列 “” を代入する命令である。2 行目の命令は、整数型変数の `i` に、`i` の初期値を示す変数 `min` を代入する命令である。3 行目の命令は、`i` と、`i` の最終値を示す `max` とを比較し、`i` が `max` 以上の場合に、基本ブロック 8 2 0 を実行し、`i` が `max` より小さい場合に、基本ブロック 8 1 0 を実行する分岐命令である。

【0 0 4 5】

4 行目の命令は、可変文字列変数を含む `StringBuffer` オブジェクト `t` のメモリ上の領域を、新たに確保する命令である。5 行目の命令は、`s` に格納されている文字列を `t` に追加する命令である。6 行目の命令は、文字列型配列変数 `A` における、`i` 番目の文字列を、`t` に追加する命令である。7 行目の命令は、`t` を、不変文字列変数の `s` に変換する不変命令である。8 行目の命令は、`i` をインクリメントする命令である。9 行目の命令は、`i` と、`i` の最終値を示す `max` とを比較し、`i` が `max` 以上の場合に、基本ブロック 8 2 0 を実行し、`i` が `max` より小さい場合に、再度基本ブロック 8 1 0 を実行する分岐命令である。

【0 0 4 6】

1 0 行目の命令は、`s` に格納されている文字列を、標準出力、例えば、ディスプレイ画面のコンソールウィンドウに出力する命令である。

【0 0 4 7】

図 9 は、意味復元部 6 0 0 が可変命令を検出したプログラムの一例を示す。意味復元部 6 0 0 は、図 8 における 4 行目及び 5 行目の命令、即ち、可変文字列変数として用いられる記憶領域を確保する命令と、可変文字列変数に不変文字列変数を複写する命令との組合せを、可変命令として検出する。そして、意味復元部 6 0 0 は、図 9 の 4 行目に示したように、検出した可変命令を、`s` から `t` への可変命令である旨を示す擬似命令に置き換える。即ち、意味復元部 6 0 0

は、可変化命令の処理を構成する複数の命令の組合せを検出し、当該複数の命令が示す処理の意味を復元する。

【 0 0 4 8 】

なお、意味復元部 6 0 0 は、プログラムにおいて、連続して配置された 4 行目及び 5 行目の命令を検出することにより、可変化命令を検出する。これに代えて、意味復元部 6 0 0 は、4 行目及び 5 行目の命令の間に他の命令が配置されている場合であっても、新たに確保された可変文字列変数に対する最初の操作が不変文字列の追加であれば、4 行目及び 5 行目の命令を、不変化命令として検出してもよい。

【 0 0 4 9 】

図 1 0 は、部分冗長除去部 6 1 0 が部分冗長性を除去したプログラムの一例を示す。部分冗長除去部 6 1 0 は、意味復元部 6 0 0 により検出された可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動するべく、4 行目の可変化命令を、基本ブロック 8 1 0 から除去すると共に、基本ブロック 8 3 0 及び基本ブロック 8 4 0 に挿入する。なお、より詳細には、部分冗長除去部 6 1 0 は、部分冗長性の除去処理として、非特許文献 2 に示した技術等と略同一の処理を行ってもよい。例えば、既存の技術によれば、部分冗長除去部 6 1 0 は、可変化命令を、当該可変化命令による変換元である変数 s に値を代入する命令より先に実行させないという条件の元、可変化命令をより先に移動する。

【 0 0 5 0 】

図 1 1 は、命令除去部 6 3 0 が可変化命令を除去したプログラムの一例を示す。代数簡約部 6 3 4 は、代数簡約的手法により、基本ブロック 8 4 0 に移動した可変化命令を削除する。即ち、基本ブロック 8 1 0 から基本ブロック 8 4 0 に至る制御フローにおいて、図 1 0 における 6 行目の不変化命令及び 1 0 行目の可変化命令の間に実行される命令が、可変文字列変数 t に格納された文字列を書き換えず、かつ可変化命令から可変化命令により変換された可変文字列変数の使用の間に実行される命令が、不変化命令の変換元の可変文字列変数と、可変化命令により変換された可変文字列変数の何れも書き換えないので、代数簡約部 6 3 4 は

、基本ブロック 8 4 0 における可変化命令を除去する。そして、代数簡約部 6 3 4 は、可変化命令より後で、不変化命令の変換元となる可変文字列変数を、可変化命令により変換された可変文字列変数として使用させる（1 0 行目の命令）。

【 0 0 5 1 】

より詳細には、代数簡約部 6 3 4 は、変数に対する値の定義と、変数に格納された値の使用との関係の解析結果、例えば、UD、DU チェインを用いて、可変文字列変数 t に格納された文字列が書き換わるか否かを判断する。更に、代数簡約部 6 3 4 は、可変文字列変数 t が、他のスレッドからアクセスされないことを更に条件として、可変文字列変数 t に格納された文字列が書き換わらないと判断する。一例としては、代数簡約部 6 3 4 は、プログラムが単一のスレッドにより実行されることを条件としてもよいし、可変文字列変数 t へのポインタを他のスレッドに通知していないことをエスケープアナリシスにより解析してもよい。

【 0 0 5 2 】

図 1 2 は、命令除去部 6 3 0 が不変化命令を除去したプログラムの一例を示す。部分不用代入文除去部 6 3 8 は、部分不用代入文の除去処理として、以下の処理を行う。部分不用代入文除去部 6 3 8 は、図 1 1 の 6 行目の不変化命令を、不変化命令より後に実行される分岐命令、例えば 7 行目の命令の分岐先のそれぞれにおいて実行させる。即ち、部分不用代入文除去部 6 3 8 は、図 1 1 の 6 行目の不変化命令を、基本ブロック 8 1 0 から除去すると共に、基本ブロック 8 4 0 及び基本ブロック 8 5 0 に移動する。そして、部分不用代入文除去部 6 3 8 は、基本ブロック 8 4 0 に移動した不変化命令の変換先である不変文字列変数 s に格納されている文字列が、基本ブロック 8 4 0 以降に実行される部分において参照されないので、基本ブロック 8 4 0 に移動した当該不変化命令を除去する。

【 0 0 5 3 】

なお、より詳細には、部分不用代入文除去部 6 3 8 は、部分不用代入文の除去処理として、非特許文献 1 に示した技術等と略同一の処理を行ってもよい。例えば、部分不用代入文除去部 6 3 8 は、不変化命令を、当該不変化命令による変換先である変数 s を参照する命令より後に実行させないという条件の元、不変化命令をより後に移動する。

【 0 0 5 4 】

図 1 3 は、コンパイラ装置 1 0 又はコンパイラ装置 6 0 のハードウェア構成の一例を示す。第 1 の実施形態又は変形例に係るコンパイラ装置 1 0 並びに第 2 の実施形態に係るコンパイラ装置 6 0 は、ホストコントローラ 1 0 8 2 により相互に接続される CPU 1 0 0 0、RAM 1 0 2 0、グラフィックコントローラ 1 0 7 5、及び表示装置 1 0 8 0 を有する CPU 周辺部と、入出力コントローラ 1 0 8 4 によりホストコントローラ 1 0 8 2 に接続される通信インターフェイス 1 0 3 0、ハードディスクドライブ 1 0 4 0、及び CD-ROM ドライブ 1 0 6 0 を有する入出力部と、入出力コントローラ 1 0 8 4 に接続される ROM 1 0 1 0、フレキシブルディスクドライブ 1 0 5 0、及び入出力チップ 1 0 7 0 を有するレガシー入出力部とを備える。

【 0 0 5 5 】

ホストコントローラ 1 0 8 2 は、RAM 1 0 2 0 と、高い転送レートで RAM 1 0 2 0 をアクセスする CPU 1 0 0 0 及びグラフィックコントローラ 1 0 7 5 とを接続する。CPU 1 0 0 0 は、ROM 1 0 1 0 及び RAM 1 0 2 0 に格納されたコンパイラプログラムに基づいて動作し、各部の制御を行う。グラフィックコントローラ 1 0 7 5 は、CPU 1 0 0 0 等が RAM 1 0 2 0 内に設けたフレームバッファ上に生成する画像データを取得し、表示装置 1 0 8 0 上に表示させる。これに代えて、グラフィックコントローラ 1 0 7 5 は、CPU 1 0 0 0 等が生成する画像データを格納するフレームバッファを、内部に含んでもよい。

【 0 0 5 6 】

入出力コントローラ 1 0 8 4 は、ホストコントローラ 1 0 8 2 と、比較的高速な入出力装置である通信インターフェイス 1 0 3 0、ハードディスクドライブ 1 0 4 0、及び CD-ROM ドライブ 1 0 6 0 を接続する。通信インターフェイス 1 0 3 0 は、ネットワークを介して他の装置と通信する。ハードディスクドライブ 1 0 4 0 は、コンパイラ装置 1 0 又はコンパイラ装置 6 0 が使用するコンパイラプログラム及びデータを格納する。CD-ROM ドライブ 1 0 6 0 は、CD-ROM 1 0 9 5 からコンパイラプログラム又はデータを読み取り、RAM 1 0 2 0 を介して入出力チップ 1 0 7 0 に提供する。

【 0 0 5 7 】

また、入出力コントローラ 1 0 8 4 には、ROM 1 0 1 0 と、フレキシブルディスクドライブ 1 0 5 0 や入出力チップ 1 0 7 0 等の比較的低速な入出力装置とが接続される。ROM 1 0 1 0 は、コンパイラ装置 1 0 又はコンパイラ装置 6 0 の起動時に CPU 1 0 0 0 が実行するブートプログラムや、コンパイラ装置 1 0 又はコンパイラ装置 6 0 のハードウェアに依存するプログラム等を格納する。フレキシブルディスクドライブ 1 0 5 0 は、フレキシブルディスク 1 0 9 0 からコンパイラプログラム又はデータを読み取り、RAM 1 0 2 0 を介して入出力チップ 1 0 7 0 に提供する。入出力チップ 1 0 7 0 は、フレキシブルディスク 1 0 9 0 や、例えばパラレルポート、シリアルポート、キーボードポート、マウスポート等を介して各種の入出力装置を接続する。

【 0 0 5 8 】

コンパイラ装置 1 0 又はコンパイラ装置 6 0 に提供されるコンパイラプログラムは、フレキシブルディスク 1 0 9 0、CD-ROM 1 0 9 5、又は IC カード等の記録媒体に格納されて利用者によって提供される。コンパイラプログラムは、記録媒体から読み出され、入出力チップ 1 0 7 0 を介してコンパイラ装置 1 0 又はコンパイラ装置 6 0 にインストールされ、コンパイラ装置 1 0 又はコンパイラ装置 6 0 において実行される。

【 0 0 5 9 】

コンパイラ装置 1 0 又はコンパイラ装置 6 0 にインストールされて実行されるコンパイラプログラムは、追加命令検出モジュールと、格納コード生成モジュールと、参照命令検出モジュールと、追加コード生成モジュールと、不変化命令検出モジュールと、部分冗長除去モジュールと、可変化命令検出モジュールと、命令除去モジュールと、代数簡約モジュールと、部分不用代入文除去モジュールと、逆意味復元モジュールと、計測コード付きプログラム実行モジュールと、可変文字列変数初期長指定モジュールとを含む。各モジュールがコンパイラ装置 1 0 又はコンパイラ装置 6 0 に働きかけて行わせる動作は、図 1 から図 1 2 において説明したコンパイラ装置 1 0 又はコンパイラ装置 6 0 における、対応する部材の動作と同一であるから、説明を省略する。

【 0 0 6 0 】

以上に示したコンパイラプログラム又はモジュールは、外部の記憶媒体に格納されてもよい。記憶媒体としては、フレキシブルディスク 1 0 9 0、CD-ROM 1 0 9 5 の他に、DVD や PD 等の光学記録媒体、MD 等の光磁気記録媒体、テープ媒体、IC カード等の半導体メモリ等を用いることができる。また、専用通信ネットワークやインターネットに接続されたサーバシステムに設けたハードディスク又は RAM 等の記憶装置を記録媒体として使用し、ネットワークを介してコンパイラプログラムをコンパイラ装置 1 0 又はコンパイラ装置 6 0 に提供してもよい。

【 0 0 6 1 】

以上の説明から明らかなように、コンパイラ装置は、文字列を追加する命令を最適化することができる。

【 0 0 6 2 】

以上、本発明を実施形態を用いて説明したが、本発明の技術的範囲は上記実施形態に記載の範囲には限定されない。上記実施形態に、多様な変更または改良を加えることができる。そのような変更または改良を加えた形態も本発明の技術的範囲に含まれ得ることが、特許請求の範囲の記載から明らかである。例えば、コンパイラ装置 1 0 は、コンパイラ装置 6 0 の各部材を更に備えてもよい。この場合、コンパイラ装置 1 0 は、予め定められた条件に基づき、第 1 実施形態及び第 2 実施形態に示した機能の何れかを選択して処理してもよい。例えば、コンパイラ装置 1 0 は、計測コード付きプログラム実行部 6 4 2 により文字列の長さが計測できた場合は、第 2 実施形態に示した処理を行い、計測できなかった場合は、第 1 実施形態に示した処理を行ってもよい。また、コンパイラ装置 1 0 は、計測コード付きプログラム実行部 6 4 2 により計測された複数の文字列の長さの分散値が、所定値より大きい場合に、第 1 実施形態に示した処理を行ってもよい。

【 0 0 6 3 】

以上で示した全ての実施形態及び変形例によると、以下の各項目に示すコンパイラ装置、コンパイラプログラム、及び記録媒体が実現される。

【 0 0 6 4 】

(項目 1) 文字列を操作するプログラムを最適化するコンパイラ装置であって、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

(項目 2) 前記複数の追加命令によって文字列を追加した後に初めて前記文字列変数を参照する参照命令を検出する参照命令検出部を更に備え、前記追加コード生成部は、前記追加コードを、全ての前記格納コードより後、かつ前記参照命令より先に実行される個所に生成する項目 1 記載のコンパイラ装置。

【 0 0 6 5 】

(項目 3) 前記追加命令検出部は、前記追加命令として、文字列を追加する処理が許可されていない不変文字列変数を、文字列を追加する処理が許可された可変文字列変数に変換する命令と、前記可変文字列変数に前記追加文字列を追加する命令と、前記可変文字列変数を不変文字列変数に変換する命令との組合せを検出する項目 1 記載のコンパイラ装置。

(項目 4) 文字列を操作するプログラムを最適化するコンパイラ装置であって、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

【 0 0 6 6 】

(項目 5) 文字列を操作するプログラムを最適化するコンパイラ装置であって、文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変化命令を検出する不変化命令検出部と、前記不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部と、前記不変化命令から前記可変化命令までの間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部とを備えるコンパイラ装置。

(項目 6) 前記不変文字列変数に格納された文字列が参照されない場合に、前記命令除去部は、前記不変化命令を更に除去する項目 5 記載のコンパイラ装置。

(項目 7) 前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う項目 6 記載のコンパイラ装置。

【 0 0 6 7 】

(項目 8) 前記可変化命令検出部は、可変文字列変数として用いられる記憶領域を確保する命令と、当該可変文字列変数に、前記不変文字列変数に格納された文字列を追加する命令との組合せを、前記可変化命令として検出する項目 5 記載のコンパイラ装置。

(項目 9) 前記可変化命令検出部により検出された前記可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う部分冗長除去部を更に備え、前記部分冗長性の前記除去処理が実行されたプログラムにおいて、前記不変化命令及び前記可変化命令の間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文

字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記命令除去部は、前記可変化命令を除去する項目 5 記載のコンパイラ装置。

【 0 0 6 8 】

(項目 1 0) 前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う項目 9 記載のコンパイラ装置。

(項目 1 1) 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部として機能させるコンパイラプログラム。

【 0 0 6 9 】

(項目 1 2) 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加

文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部として機能させるコンパイラプログラム。

(項目 1 3) 文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変命令を検出する不変命令検出部と、前記不変文字列変数を可変文字列変数に変換する可変命令を検出する可変命令検出部と、前記不変命令から前記可変命令までの間に実行される命令が、前記不変命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変命令から前記可変命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変命令の変換元の可変文字列変数と、前記可変命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変命令を除去し、前記可変命令より後で、前記不変命令の変換元となる可変文字列変数を、前記可変命令により変換された可変文字列変数として使用させる命令除去部として機能させるコンパイラプログラム。

(項目 1 4) 項目 1 1 から 1 3 の何れかに記載のコンパイラプログラムを記録した記録媒体。

【0 0 7 0】

【発明の効果】

上記説明から明らかなように、本発明によれば文字列を操作するプログラムを最適化することができる。

【図面の簡単な説明】

【図 1】

図 1 は、コンパイラ装置 1 0 のブロック図を示す。

【図 2】

図 2 は、コンパイラ装置 1 0 の動作フローを示す。

【図 3】

図 3 (a) は、最適化対象のプログラムのソースコードの一例を示す。

図 3 (b) は、プログラムがコンパイルされた結果を示す。

図 3 (c) は、コンパイラ装置 1 0 による最適化を終えたプログラムの一例を示す。

【図 4】

図 4 は、変形例において、コンパイラ装置 1 0 がプログラムを最適化する一例を示す。

【図 5】

図 5 (a) は、第 1 の他の方法においてプログラムが最適化された結果を示す。

図 5 (b) は、第 2 の他の方法においてプログラムが最適化された結果を示す。

【図 6】

図 6 は、コンパイラ装置 6 0 のブロック図を示す。

【図 7】

図 7 は、コンパイラ装置 6 0 の動作フローを示す。

【図 8】

図 8 は、コンパイラ装置 6 0 が最適化する対象のプログラムの一例を示す。

【図 9】

図 9 は、意味復元部 6 0 0 が可変化命令を検出したプログラムの一例を示す。

【図 1 0】

図 1 0 は、部分冗長除去部 6 1 0 が部分冗長性を除去したプログラムの一例を示す。

【図 1 1】

図 1 1 は、命令除去部 6 3 0 が可変化命令を除去したプログラムの一例を示す。

【図 1 2】

図 1 2 は、命令除去部 6 3 0 が不変化命令を除去したプログラムの一例を示す。

【図 1 3】

図 1 3 は、コンパイラ装置 1 0 又はコンパイラ装置 6 0 のハードウェア構成の

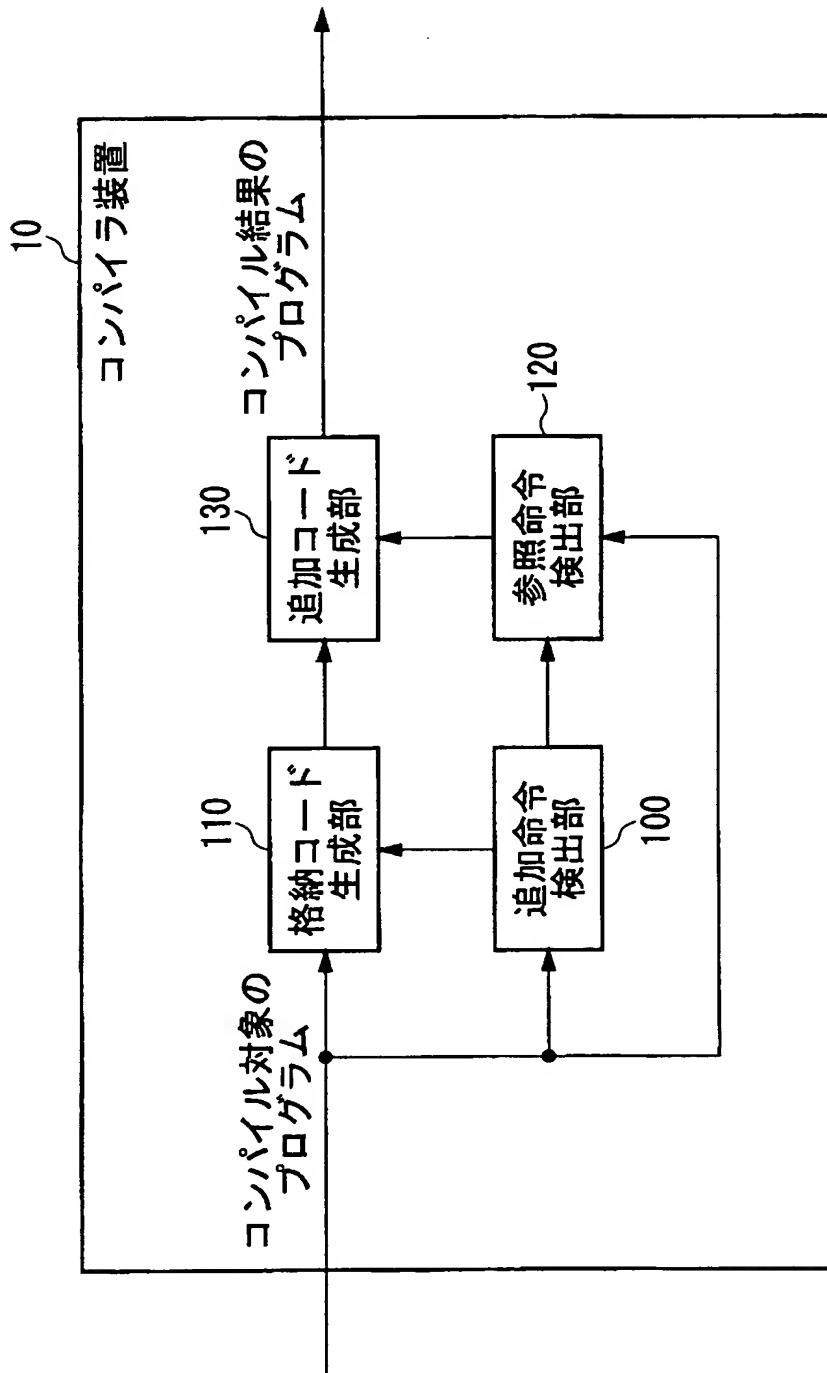
一例を示す。

【符号の説明】

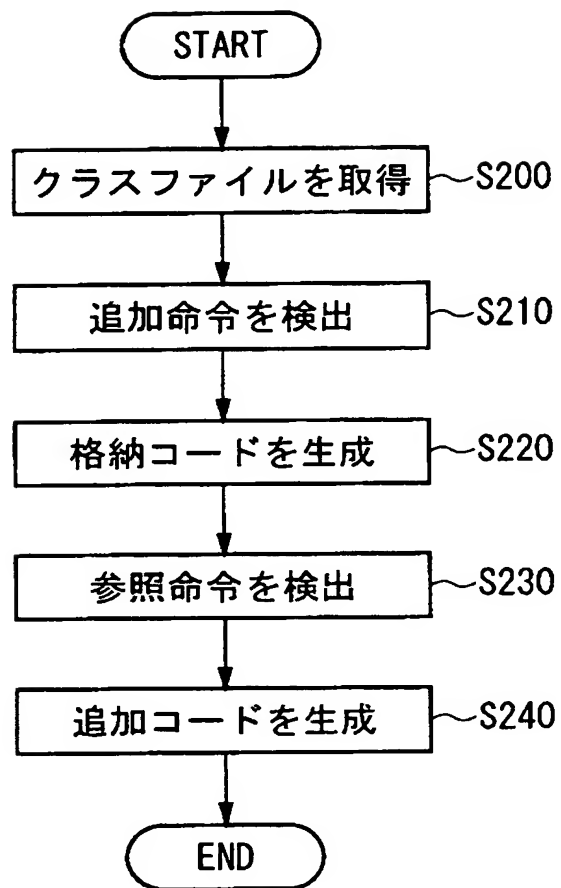
- 1 0 コンパイラ装置
- 6 0 コンパイラ装置
- 1 0 0 追加命令検出部
- 1 1 0 格納コード生成部
- 1 2 0 参照命令検出部
- 1 3 0 追加コード生成部
- 6 0 0 可変化命令検出部
- 6 1 0 意味復元部
- 6 2 0 不変化命令検出部
- 6 3 0 命令除去部
- 6 3 4 代数簡約部
- 6 3 8 部分不用代入文除去部
- 6 4 0 逆意味復元部
- 6 4 2 計測コード付きプログラム実行部
- 6 4 5 不変文字列変数最終長データベース
- 6 5 0 可変文字列変数初期長指定部
- 8 0 0 基本ブロック
- 8 1 0 基本ブロック
- 8 2 0 基本ブロック
- 8 3 0 基本ブロック
- 8 4 0 基本ブロック
- 8 5 0 基本ブロック

【書類名】 図面

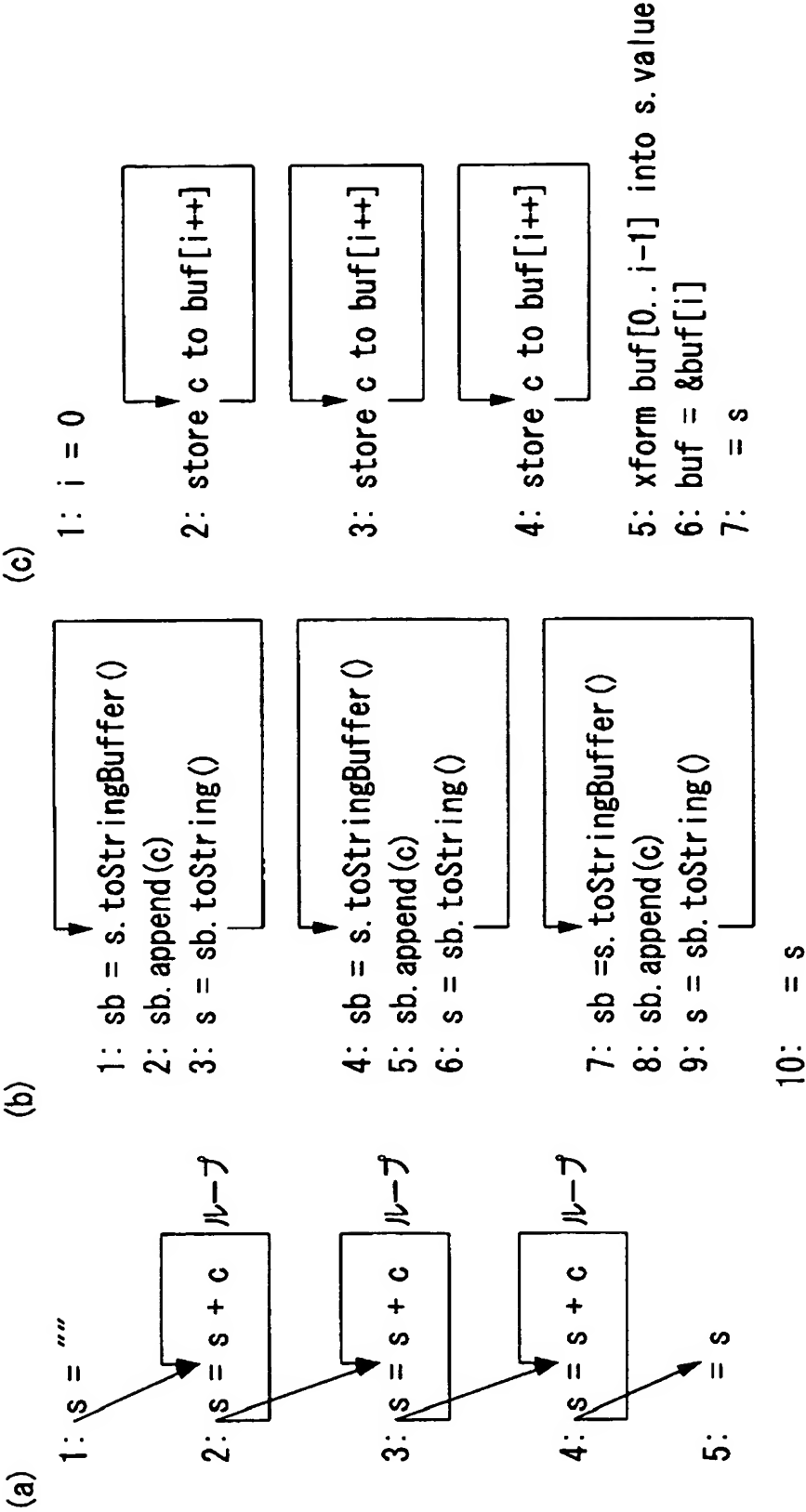
【図 1】



【図 2】



【図 3】



【図 4】

1: bufの初期化

2: store c to buf[]
3: buf = buf → next

4: store c to buf[]
5: buf = buf → next

6: store c to buf[]
7: buf = buf → next

8: bufのリストをs.valueに変換

9: = s

【図 5】

(a)

1: sb = s.toStringBuffer()

2: sb.append(c)

3: sb.append(c)

4: sb.append(c)

5: s = sb.toString()

6: = s

(b)

1: s =

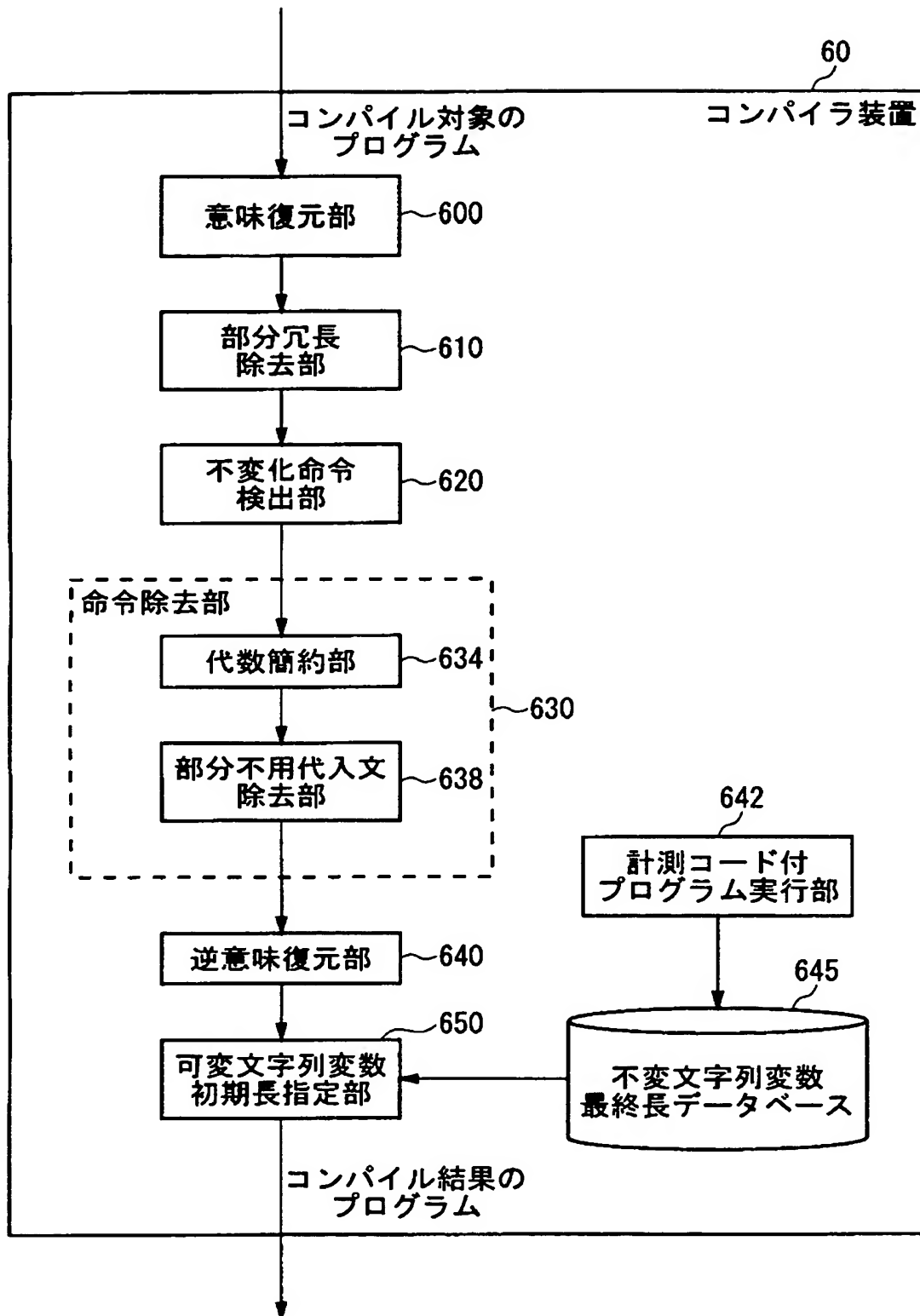
2: s = new String(s, c)

3: s = new String(s, c)

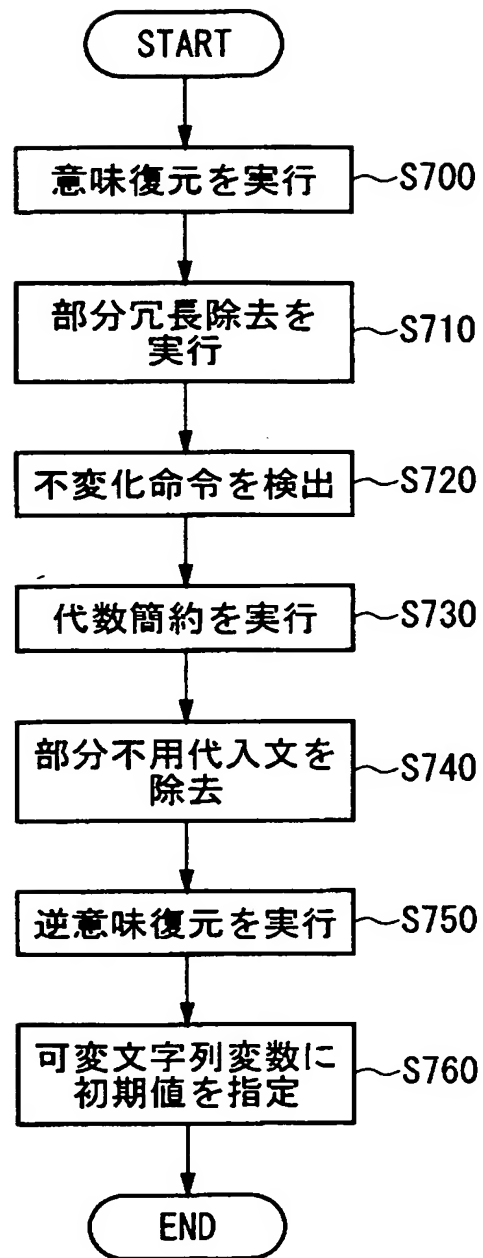
4: s = new String(s, c)

5: = s

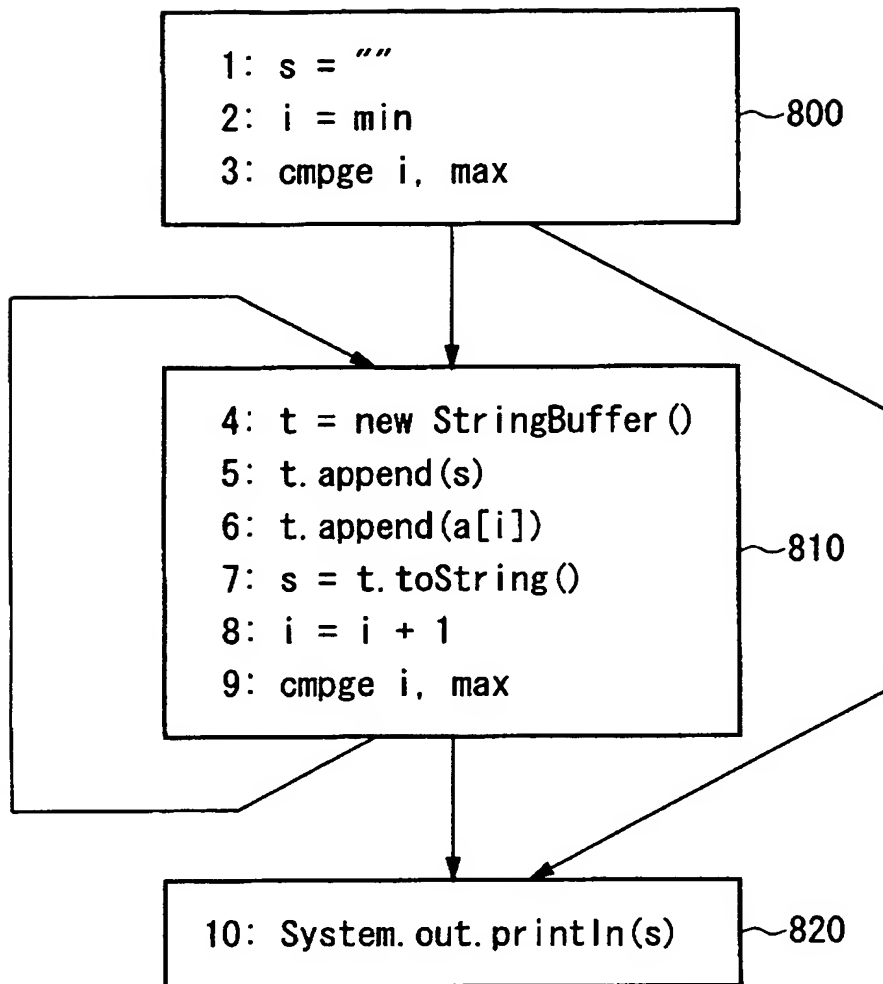
【図 6】



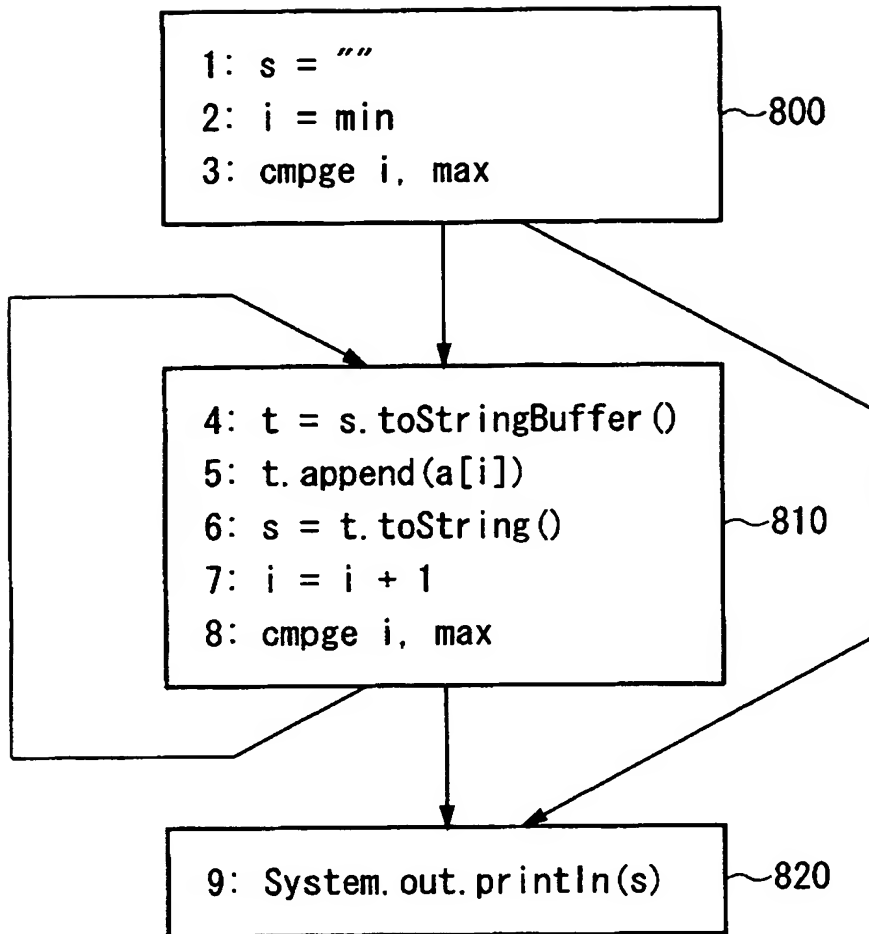
【図 7】



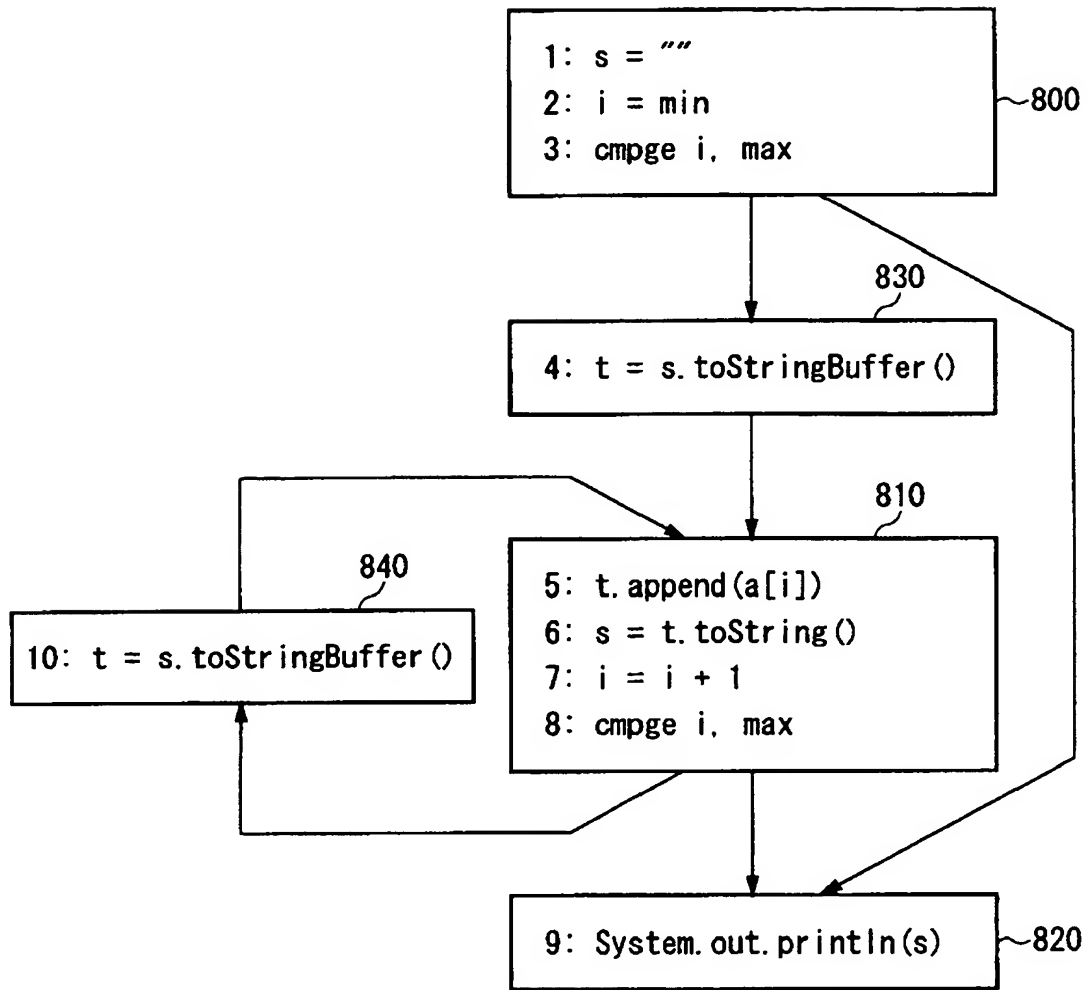
【図 8】



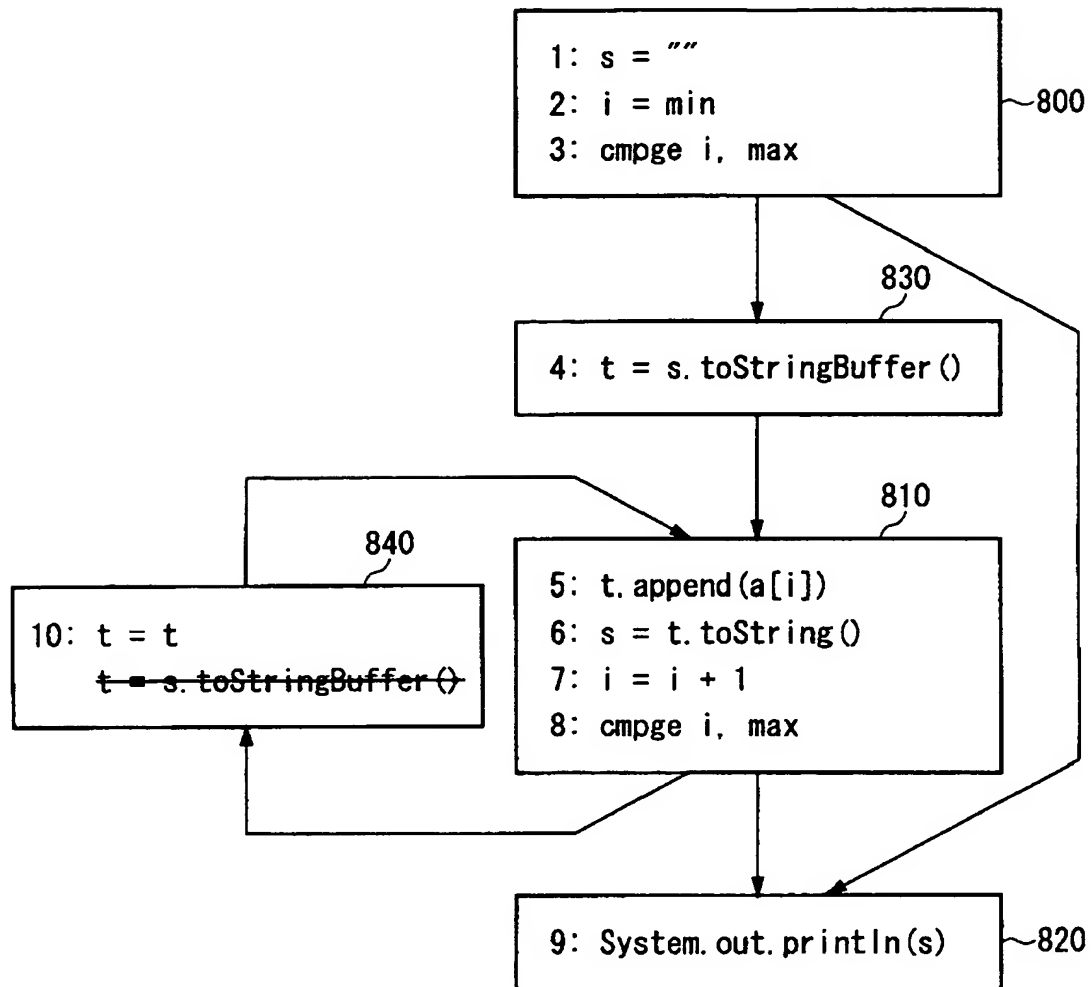
【図 9】



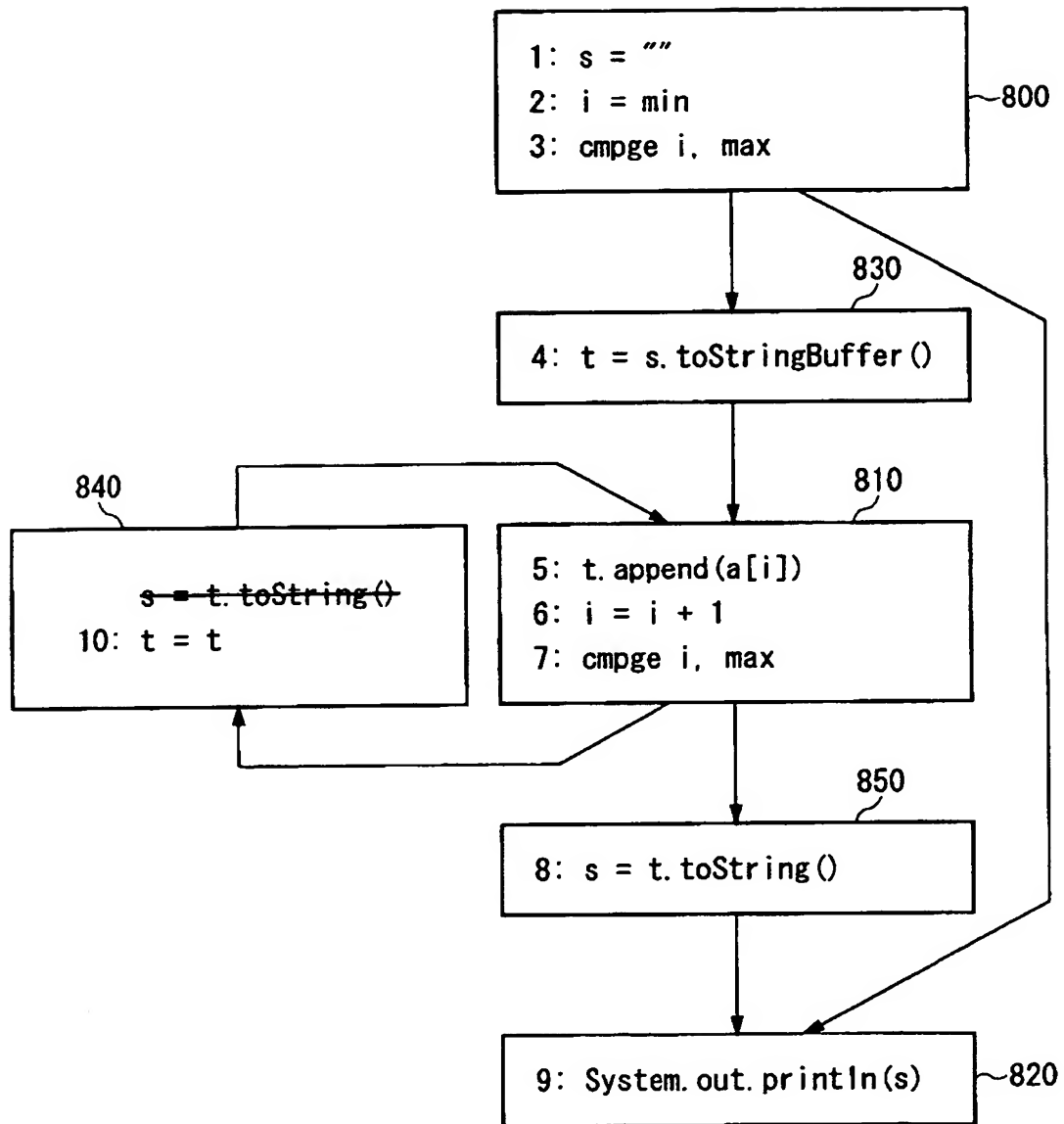
【図 1 0】



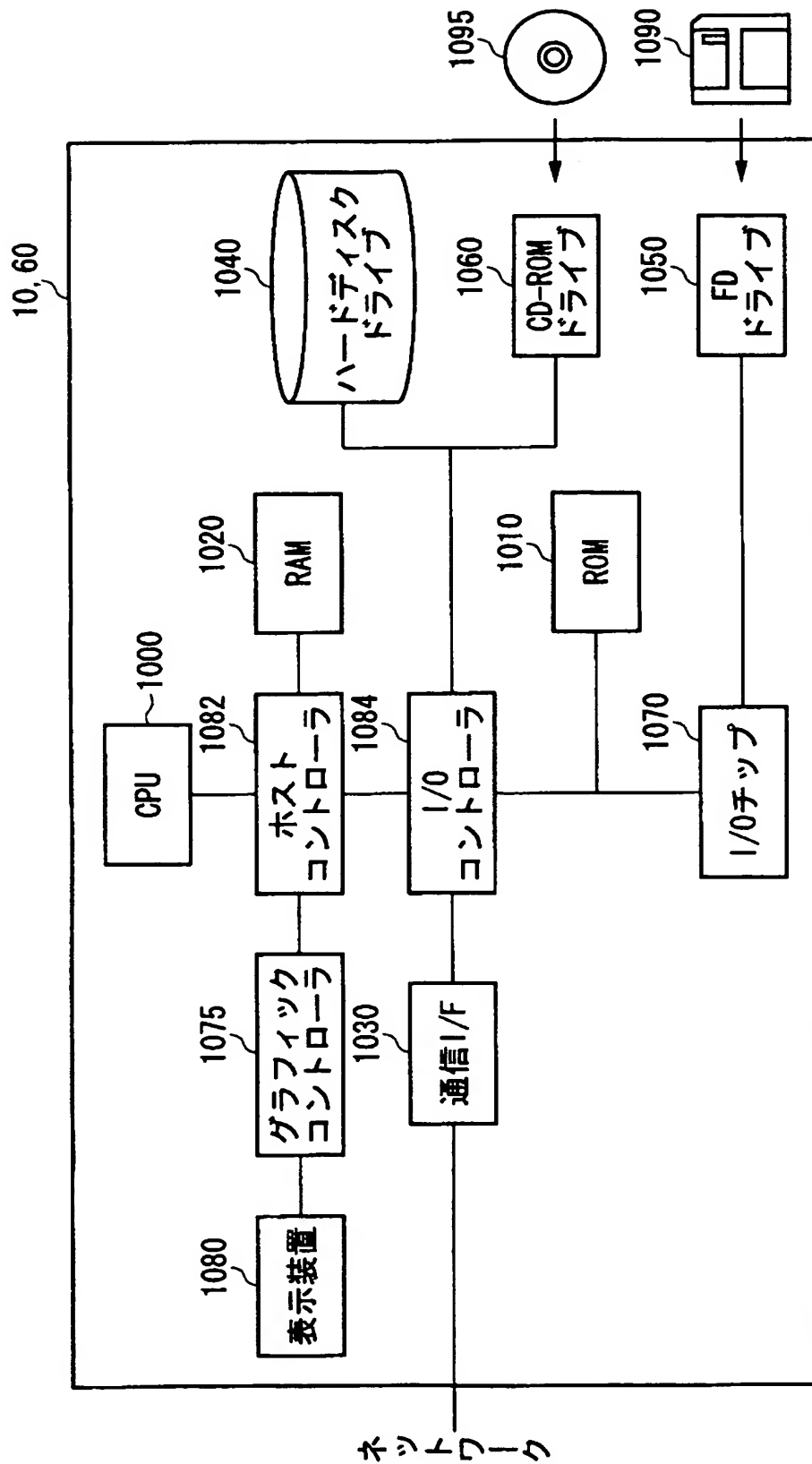
【図 11】



【図 12】



【図 13】



【書類名】 要約書

【要約】

【課題】 文字列を操作するプログラムを最適化する。

【解決手段】 文字列を操作するプログラムを最適化するコンパイラ装置は、プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、プログラムにおける、文字列変数を参照する命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する追加コード生成部とを備える。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願 2 0 0 3 - 0 4 9 4 1 4
受付番号	5 0 3 0 0 3 1 0 5 3 0
書類名	特許願
担当官	伊藤 雅美 2 1 3 2
作成日	平成 1 5 年 4 月 8 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	390009531
【住所又は居所】	アメリカ合衆国 1 0 5 0 4、ニューヨーク州 アーモンク ニュー オーチャード ロード
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博

【代理人】

【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

【代理人】

【識別番号】	100108501
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番 1 4 日本アイ・ビー・エム株式会社 知的所有権
【氏名又は名称】	上野 剛史

【復代理人】

申請人	
【識別番号】	100104156
【住所又は居所】	東京都新宿区新宿 1 丁目 2 4 番 1 2 号 東信ビル 6 階 龍華国際特許事務所
【氏名又は名称】	龍華 明裕

次頁無

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 2002年 6月 3日

[変更理由] 住所変更

住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク ニ
ュー オーチャード ロード

氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーショ
ン